Дискретная математика

Логические схемы

Константин Чухарев

Осень 2026

Системы счисления

> *Что я не могу создать, я не понимаю.*
>
> ***— Ричард Фейнман***

Как записать число

Компьютер хранит всё в нулях и единицах. Чтобы складывать числа в железе, их записывают двоично — как именно, задаёт система счисления.

Определение 1 (Позиционная система)

В системе с основанием $b \geq 2$ число $d_k \ldots d_0$ (где $0 \leq d_i < b$) представляет

$$\sum_{i=0}^{k} d_i b^i.$$

- При $b = 2$ — двоичная.
- При $b = 16$ — шестнадцатеричная.

Примеры записи

Пример

$1011_2 = 1 \cdot 8 + 1 \cdot 2 + 1 = 11_{10}$.

$11111111_2 = 255_{10}$ — максимум байта.

Десятичная система — историческая случайность. Двоичная — родной язык схем.

Перевод между системами

Пример

Число 13 в двоичной записи — это 1101_2: делим на 2 с остатком, остатки читаем снизу вверх.

$\frac{13}{2} = 6$, остаток 1. $\frac{6}{2} = 3$, остаток 0.
$\frac{3}{2} = 1$, остаток 1. $\frac{1}{2} = 0$, остаток 1.

Читаем снизу вверх: 1101_2.

Одна шестнадцатеричная цифра кодирует 4 бита: $\mathrm{BE}_{16} = 1011\ 1110_2$.

Цвета в HTML: `#FF00FF` — пурпурный.

Разряды — гирьки с весами $1, 2, 4, 8, \ldots$.

Бит решает, включена гирька в сумму или нет.

Вентили и схемы

Логические вентили

Вентиль	Функция	Смысл
NOT	$\neg x$	инвертирует
AND	$x \wedge y$	оба входа 1
OR	$x \vee y$	хотя бы один 1
NAND	$\neg(x \wedge y)$	отрицание AND
NOR	$\neg(x \vee y)$	отрицание OR
XOR	$x \oplus y$	входы различны

Функциональная полнота

- NAND и NOR функционально полны.
- XOR незаменим: различает биты, контролирует паритет.
- В CMOS вентиль NAND собирают из четырёх транзисторов: два NMOS последовательно тянут выход вниз, два PMOS параллельно — вверх, поэтому схема не тратит ток в покое.

Схемы как DAG

Выходы одних вентилей подаются на входы других.

Определение 2 (Логическая схема)

Схема — ориентированный ациклический граф:

- истоки — это входы,
- внутренние узлы — вентили (*gates*),
- стоки — выходы.

Ацикличность — комбинационная логика (*combinational logic*): результат зависит только от текущих входов.

Пример схемы

Пример

$(x \wedge y) \vee \neg z$: AND, NOT, OR — три вентиля.

Глубина 2: AND и NOT работают параллельно, OR ждёт оба.

- Таблица истинности задаёт, **что** вычисляется, а схема — **как**.
- Если в схеме есть цикл, её значение перестаёт быть определённым.

Размер и глубина

Определение 3

Размер схемы — число вентилей (площадь, стоимость).

Глубина — длиннейший путь от входа к выходу (задержка, частота).

Выбор между размером и глубиной — базовый инженерный компромисс.

Схему с малой глубиной приходится собирать из большего числа вентилей, а компактная схема работает медленнее.

Пример

$x \oplus y = (x \vee y) \wedge \neg(x \wedge y)$.

Размер 4, глубина 3.

Размер и глубина [2]

Задержку даёт критический путь — не сумма всех вентилей.

Семейства схем

Одна схема имеет фиксированное число входов.

Схема для 4-битного сложения не сложит 8-битные числа.

Определение 4

Семейство схем — последовательность $C_1, C_2, \ldots$, где C_n имеет n входов.

Семейство **равномерно**, если существует алгоритм, строящий C_n по n.

Равномерные семейства

Пример

Чётность: C_n — $n - 1$ XOR-вентилей.

Равномерные полиномиальные семейства схем — это в точности класс P (машину Тьюринга — модель алгоритма — определим в лекции о вычислимости).

Схема — не программа: она не параметризована размером входа.

Поэтому нужен формализм семейств.

Сумматоры

Сложение столбиком

Алгоритм сложения не зависит от основания.

В двоичной: $1 + 1 = 10_2$ — ноль и перенос.

Уметь нужно немногое:

- сложить столбец,
- передать перенос соседнему.

Процессор не выполняет арифметику напрямую: сложение собирается из логических вентилей.

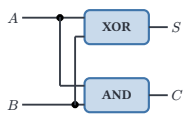
Полусумматор

Определение 5 (Полусумматор (*half adder*))

Складывает два бита:

$$S = A \oplus B, \quad C = A \wedge B.$$

Формирует перенос, но не принимает входящий.



Полусумматор: таблица истинности

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

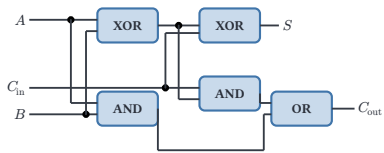
«Полу-» — честно: колонка складывает три бита, а эта схема — только два.

Полный сумматор

Определение 6 (Полный сумматор (*full adder*))

Складывает три бита: A, B и входящий перенос C_{in}:

$$S = A \oplus B \oplus C_{\text{in}}, \quad C_{\text{out}} = (A \wedge B) \vee (C_{\text{in}} \wedge (A \oplus B)).$$



Полный сумматор: перенос

Перенос $C_{\text{out}} = 1$, когда не менее двух из трёх входов равны 1.

Это функция большинства.

Пример

Формулы переносятся в код: сумма — XOR трёх битов, перенос — большинство.

Последовательный перенос

Определение 7 (Ripple-carry)

Цепочка из n полных сумматоров.

Перенос через все разряды: размер $O(n)$, глубина $O(n)$.

Пример

$0111 + 0001$: перенос идёт по всем четырём разрядам.

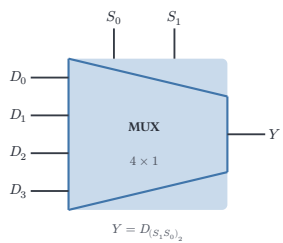
Результат 1000 — каждый разряд ждёт переноса.

- Глубина ограничивает тактовую частоту.
- Опережающий перенос (*carry-lookahead*): глубина $O(\log n)$, размер больше.
- Вычитание — сложение с дополнительным кодом (*two's complement*): в n-битной арифметике $A - B = A + \left(\overline{B} + 1\right) (\mathrm{mod}\, 2^n)$.

АЛУ

АЛУ — вычислительное ядро процессора.

- **Мультиплексор** «из 2^k в 1» — селектор, который по k адресным битам выбирает один из входов.
- **Декодер** k-в-2^k — активирует один из выходов.
- Однобитный срез АЛУ: параллельные блоки AND, OR, ADD, SUB, мультиплексор выбирает.



АЛУ: срезы и состав

n-битное АЛУ — n срезов с цепочкой переносов.

Современные АЛУ:

- умножители,
- сдвигатели,
- FPU.

Код Грея

Код Грея

При переключении нескольких битов сразу возникают ложные состояния.

Определение 8 (Код Грея)

Упорядочение всех 2^n двоичных строк длины n, где любые две соседние строки, включая последнюю и первую, отличаются ровно в одном бите.

Пример

G_3: 000, 001, 011, 010, 110, 111, 101, 100.

Каждая соседняя пара отличается в одном бите.

Последняя и первая — тоже: код цикличен.

Энкодеры: один бит меняется за раз — нет ложных промежуточных состояний.

Построение

Теорема 1 (Рекурсивное построение)

$G_1 = [0, 1]$.

$$G_{n+1} = 0G_n, 1G_n^R,$$

где $0G_n$ означает каждое слово G_n с приписанным слева нулём, а $1G_n^R$ — каждое слово G_n в обратном порядке с приписанной слева единицей.

Пример и цикличность

Пример

$G_2 = [00, 01, 11, 10]$.

$G_3 = [000, 001, 011, 010, 110, 111, 101, 100]$.

Стык блоков отличается только первым битом.

Обращение второй половины и даёт цикличность: последнее слово первой половины и первое слово второй отличаются только первым битом.

Преобразование

Теорема 2 (Двоичный и код Грея)

Двоичный в Грей: $g_{n-1} = b_{n-1}$, $g_i = b_i \oplus b_{i+1}$.

Грей в двоичный: $b_{n-1} = g_{n-1}$, $b_i = g_i \oplus b_{i+1}$ — накопительный XOR от старшего бита.

Пример

$b = 1011$: $g = 1110$.

Старший бит сохраняется, дальше — XOR соседних.

Оба преобразования — за $O(n)$, цепочка XOR-вентилей.

Применения:

- энкодеры,

Преобразование [2]

- карты Карно,
- вложение в гиперкуб (гамильтонов цикл).

Что мы поняли?

- Схема — ациклический граф вентилей: таблица истинности задаёт **что** вычисляется, схема — **как**.
- Размер и глубина — два ресурса схемы. Задержку даёт критический путь, а не число вентилей.
- Полный сумматор: сумма — XOR трёх битов, перенос — большинство. Опережающий перенос снижает глубину до $O(\log n)$.
- Код Грея меняет ровно один бит за шаг, поэтому энкодеры не дают ложных промежуточных состояний.

Вопросы для самопроверки

- Переведите 1101_2 в десятичную и 13 в двоичную.
- Почему XOR не нужен для полноты, но незаменим в схемах?
- Постройте схему для $(x \wedge y) \vee \neg z$. Каковы размер и глубина?
- Почему цикл в комбинационной схеме — неопределённость?
- Запишите формулы полусумматора и полного сумматора.
- Почему ripple-carry медленный, и как ускоряет опережающий перенос?
- Постройте G_4 рекурсивно.
- Переведите 0110 из кода Грея в двоичный.